An Algorithm for Orienting Graphs Based on Cause-Effect Pairs and Its Applications to Orienting Protein Networks

Alexander Medvedovsky[1], Vineet Bafna[2], Uri Zwick[1], and Roded Sharan[1]

[1] School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel.
{medv,zwick,roded}@post.tau.ac.il.
[2] Dept. of Computer Science, UC San Diego, USA. vbafna@cs.ucsd.edu.

Abstract. We consider a graph orientation problem arising in the study of biological networks. Given an undirected graph and a list of ordered source-target pairs, the goal is to orient the graph so that a maximum number of pairs will admit a directed path from the source to the target. We show that the problem is NP-hard and hard to approximate to within a constant ratio. We then study restrictions of the problem to various graph classes, and provide an $O(\log n)$ approximation algorithm for the general case. We show that this algorithm achieves very tight approximation ratios in practice and is able to infer edge directions with high accuracy on both simulated and real network data.

1 Introduction

One of the major roles of protein-protein interaction (PPI) networks is to transmit signals within the cell in response to genetic and environmental cues. Technologies for measuring PPIs (see, e.g., [3]) do not provide information on the direction in which the signal flows. It is thus a great challenge to orient a given network by combining causal information on cellular events. One such source of information is perturbation experiments in which a gene is perturbed and as a result other genes change their expression levels.

In graph theoretic terms, one is given an undirected graph and a list of cause-effect pairs. The goal is to direct the edges of the graph, assigning a single direction to each edge, so that a maximum number of pairs admit a directed path from the cause to the effect. In fact, by contracting cycles in the graph one can easily reduce the problem to that of orienting a tree. Hakimi et al. [4] studied a restricted version of the problem where the list of vertex pairs includes all possible pairs, giving a quadratic time algorithm for it. Another variant of the problem was studied in [1] and [5], where rather than maximizing the total number of pairs, an algorithm was given to decide if one can satisfy *all* given pairs.

In this paper we study the resulting tree orientation problem. We prove that it is NP-hard and hard to approximate to within a constant ratio, study restrictions of the problem to various graph classes, and provide an $O(\log n)$

approximation algorithm for the general case, where n is the size of the tree. We show that this algorithm achieves tight approximation ratios in practice and is able to infer edge directions with high accuracy on both simulated and real network data.

The paper is organized as follows: In Section 2 the graph orientation problem is presented and its complexity is analyzed. Section 3 provides exact and approximate algorithms for restrictions of the problem, and an approximation algorithm for the general case. Biological applications of the latter algorithm are described in Section 4. For lack of space, some proofs are shortened or omitted.

2 Problem Definition

Let $G = (V, E)$ be an undirected graph. An *orientation* $\boldsymbol{G}$ of G is a directed graph obtained from G by orienting each edge $(u, v) \in E$ either from u to v or from v to u. Let $P \subseteq V \times V$ be a set of ordered source-target pairs. A pair $(a, b) \in P$ is satisfied by a given orientation $\boldsymbol{G}$ of G if there is a *directed* path from a to b in $\boldsymbol{G}$. Our goal is to find an orientation $\boldsymbol{G}$ of G that simultaneously satisfies as many pairs from P as possible.

If the graph G contains a cycle C, then it is easy to see that, for any set P, there is an optimal orientation of G in which all the edges of C are oriented in the same direction and, consequently, all pairs that connect two vertices in C are satisfied. The original problem can therefore be solved by *contracting* the cycle C and then solving an equivalent problem on the contracted graph. Thus, the interesting case is when the graph G is a tree.

Definition 1. Maximum Tree Orientation *(MTO): Given an undirected tree T and a set P of ordered pairs of vertices, find an orientation of the edges of T that maximizes the number of pairs in P that are satisfied.*

In the decision version of the problem, the input includes T, P, and an integer $k \leq |P|$, and the question is whether the edges can be directed so that at least k pairs in P are satisfied. As we show next, the problem is NP-hard even when T is a star or a binary tree.

Theorem 1. *MTO is NP-complete.*

Proof. The problem is clearly in NP. We show NP-hardness by reduction from Max Di-Cut [8], which is defined as follows: given a directed graph $G = (V, E)$ and an integer $k \leq |E|$, is there a cut $A \subset V$ such that there are at least k edges $e = (u, v)$, with $u \in A$ and $v \in V \backslash A$.

We map an instance (G, k) of Max Di-Cut into an instance $(T = (V', E'), P, k)$ of MTO in the following way: $V' = V \cup \{O\}$, $E' = \{(v, O) : v \in V\}$ and $P = E$.

Given a cut $A \subset V$ with k crossing edges, it is easy to see that each pair corresponding to such an edge can be satisfied: for all $v \in A$ direct the edge (v, O) toward O, and direct all other edges away from O.

On the other hand, suppose that we have directed the edges of T so that k pairs are satisfied. Note that if (u, v) is satisfied then u is directed toward O,

and no pair (v', u) can be satisfied. Therefore, the cut defined by $A = \{u \mid (u, O)$ is directed toward $O\}$, is of size k. ■

Corollary 1. *MTO is NP-complete even on stars.*

As Max Di-Cut is hard to approximate to within a factor of $\frac{11}{12} \simeq 0.9166$ (Håstad [6]), and the reduction is approximation preserving, we conclude:

Corollary 2. *It is NP-hard to approximate MTO to within a factor of* $\frac{11}{12}$.

Theorem 2. *MTO is NP-complete on binary trees.*

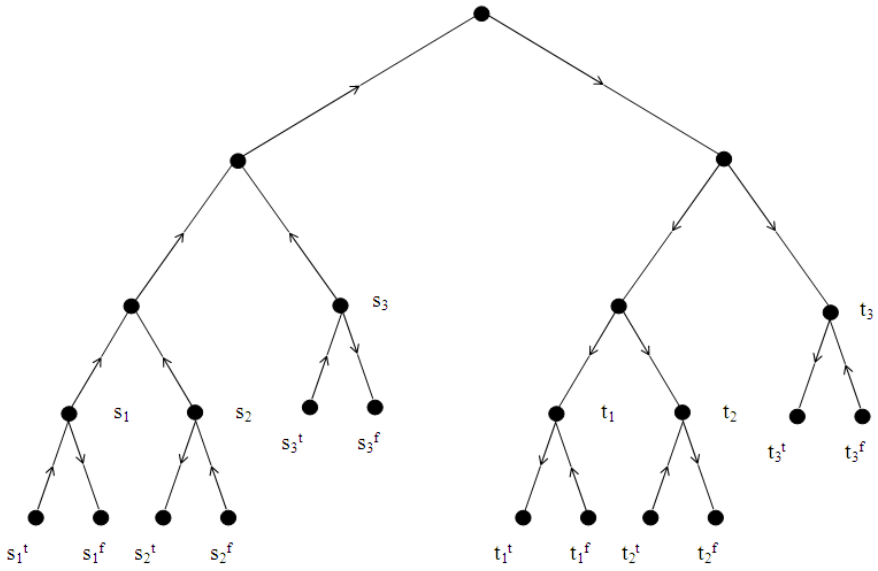

Fig. 1. An example of the reduction from MAX-2-SAT to MTO. The input 2-SAT formula is $(x_1 \vee \neg x_2) \wedge (x_3 \vee \neg x_2)$; x_1 and x_3 are assigned a *True* value, and x_2 is assigned a *False* value.

Proof. The problem is clearly in NP. We prove NP-hardness by a reduction from Max 2-SAT, where each clause is assumed to contain exactly two literals. Suppose f is a 2-SAT formula with variables $x_1, ..., x_n$. Create a binary tree T with subtrees T_s and T_t, so that T_s has a leaf s_i, and T_t has a leaf t_i for each variable x_i. Create two child nodes s_i^t and s_i^f for each s_i, and t_i^t and t_i^f for each t_i (see Figure 1).

To complete the reduction we need to specify a set of pairs to be satisfied. This set will be composed of two subsets: P_1, forcing the choice of a truth value for each variable, and P_2, relating these truth values to the clauses in f.

The truth value of a variable will be set by forcing a directed path between s_i^t and s_i^f. If the path is directed from s_i^t to s_i^f we will interpret it as assigning the value *True* to x_i; if it is directed the other way, we will associate the value *False* with x_i. To this end, for every variable x_i participating in n_i clauses, we will add $3n_i$ pairs (s_i^t, s_i^f) and $3n_i$ pairs (s_i^f, s_i^t) to P_1. Similarly, we will force a path between t_i^t and t_i^f, indicating the truth value of x_i, but this time a path from t_i^t to t_i^f will indicate *False* and the opposite direction will indicate *True*. Again, this will be done by adding $3n_i$ pairs (t_i^t, t_i^f) and $3n_i$ pairs (t_i^f, t_i^t) to P_1. Finally, to force the consistency of truth association in T_s and T_t, we will force a directed path from s_i^t to t_i^t or from s_i^f to t_i^f by adding $3n_i$ pairs (s_i^t, t_i^t) and $3n_i$ pairs (s_i^f, t_i^f) to P_1.

The complementing subset of pairs is defined as follows:

$$\begin{aligned} P_2 = &\{(s_i^t, t_j^t), (s_i^t, t_j^f), (s_i^f, t_j^t) | (x_i \vee x_j) \in f\} \cup \\ &\{(s_i^f, t_j^t), (s_i^f, t_j^f), (s_i^t, t_j^t) | (\neg x_i \vee x_j) \in f\} \cup \\ &\{(s_i^t, t_j^t), (s_i^t, t_j^f), (s_i^f, t_j^f) | (x_i \vee \neg x_j) \in f\} \cup \\ &\{(s_i^f, t_j^t), (s_i^f, t_j^f), (s_i^t, t_j^f) | (\neg x_i \vee \neg x_j) \in f\} \end{aligned}$$

Define $P = P_1 \cup P_2$. We claim that c clauses in f can be satisfied iff $18n + c$ pairs in P can be satisfied, where n is the total number of clauses in f.

Suppose that there is a truth assignment that satisfies c clauses in f. Direct the edges (s_i, s_i^t), (s_i, s_i^f), (t_i, t_i^t) and (t_i, t_i^f) according to the assignment. Direct all other edges in T_s upwards, and edges in T_t downwards. For each i, there are $9n_i$ satisfied pairs in P_1. Since $\sum_i n_i = 2n$, the number of satisfied pairs in P_1 is $18n$. Clearly, for every satisfied clause there is a satisfied pair from P_1. Thus, $18n + c$ pairs of P can be satisfied.

Conversely, suppose we have an orientation of T so that $18n + c$ pairs of P are satisfied. For each i there are at most $9n_i$ satisfied pairs in P_1. If the total number of satisfied pairs in P_1 is less than $18n$, then for some i there are less than $9n_i$ satisfied pairs (out of the ones associated with it). This implies that the directions of the edges (s_i, s_i^t), (s_i, s_i^f), (t_i, t_i^t), (t_i, t_i^f) are inconsistent. Thus, either $6n_i$, $3n_i$ or 0 of the corresponding pairs are satisfied. However, if we make these edge directions consistent, we add at least $3n_i$ satisfied pairs from P_1 and lose at most $3n_i$ pairs involving one of $s_i^t, s_i^f, t_i^t, t_i^f$ from P_2. Thus, w.l.o.g., we can assume that these edges are directed consistently, implying exactly $18n$ satisfied pairs from P_1. In addition, we have c satisfied pairs from P_2. Moreover, due to the consistency assumption, each clause can have at most one associated pair satisfied. It follows that c clauses can be satisfied in f. ■

3 Exact and Approximation Algorithms for MTO

As we have shown that MTO is NP-hard, we describe polynomial time algorithms for special cases, and approximation algorithms for special cases and for the general case. We start by providing an integer programming (IP) formulation

of the problem that will be useful for studying the practical performance of the algorithms we propose for MTO.

3.1 An Integer Program Formulation

Since every two vertices in a tree are connected by a unique path, MTO can be solved using the following integer program:

1. For each vertex pair $p \in P$ introduce a Boolean variable $y(p)$, indicating whether it is satisfied or not.
2. For each edge $e = (u, v) \in T$, where $u < v$, introduce a Boolean variable $x(e)$, indicating its direction (1 if it is directed from u to v, and 0 otherwise).
3. For each pair $p = (a, b) \in P$ and every tree edge $e = (u, v) \in T$, where $u < v$: if the path from a to b in T uses e in the direction from u to v, introduce a constraint $y(p) \leq x(e)$, and if it uses the edge in the direction from v to u, introduce a constraint $y(p) \leq 1 - x(e)$.
4. Maximize the objective function $\sum_{p \in P} y(p)$.

It is possible to consider an LP-relaxation of the above integer programming, but it is not very useful as a value of $|P|/2$ can always be obtained by setting $x(e) = y(p) = \frac{1}{2}$ for every $e \in T$ and $p \in P$.

3.2 Solving MTO on Paths

In this section we present a simple dynamic programming algorithm that solves MTO on a path in polynomial time.

Assume that the vertices on the path are numbered consecutively from 1 to n. The edges of the path are $(i, i+1)$, for $1 \leq i < n$. We think of vertex i as lying to the *left* of vertex $i+1$. We also let $[i, j] = \{i, i+1, \ldots, j\}$.

Let P be the input set of pairs. For every $1 \leq i < j \leq n$, let $v_{ij}^{+} = |\{(a, b) \in P \mid i \leq a < b \leq j\}|$ and $v_{ij}^{-} = |\{(b, a) \in P \mid i \leq a < b \leq j\}|$. In other words, v_{ij}^{+} is the number of pairs of P with both endpoints in the interval $[i, j]$ that are satisfied when the edges $(i, i+1), \ldots, (j-1, j)$ are all oriented to the right, while v_{ij}^{-} is the number of such pairs satisfied when the edges are oriented to the left. Let v_{ij} be the maximal number of pairs of P with both endpoints in $[i, j]$ that can be simultaneously satisfied using *any* orientation of the edges in the interval $[i, j]$. We claim:

Lemma 1. *For every* $1 \leq i < j \leq n$ *we have* $v_{ij} = \max\{ v_{ij}^{+}, v_{ij}^{-}, \max\limits_{i<k<j} v_{ik} + v_{kj}\}$.

Proof. The proof that $v_{ij} \geq \max\{ v_{ij}^{+}, v_{ij}^{-}, \max_{i<k<j} v_{ik} + v_{kj}\}$ is straightforward. We prove, therefore, the opposite inequality. Consider the orientation of $[i, j]$ that achieves the maximal value of v_{ij}. If all the edges in this orientation are oriented to the right, then $v_{ij} = v_{ij}^{+}$ and we are done. Similarly, if they are all oriented to the left, then $v_{ij} = v_{ij}^{-}$. Otherwise, there is a vertex $i < k < j$ for which the edges $(k-1, k)$ and $(k, k+1)$ have opposite orientations. It follows that no pair (a, b)

with $a < k < b$ or $b < k < a$ can be satisfied by such an orientation. Hence, all edges satisfied by this orientation lie in either $[i, k]$ or $[k, j]$; thus, $v_{ij} = v_{ik} + v_{kj}$, as required. ■

As an immediate corollary we get:

Theorem 3. *MTO on paths of length n can be solved in $O(n^3)$ time.*

3.3 Approximating MTO on Stars

A *star* is a tree in which the root is directly connected to all the leaves (i.e., a tree with one level). In this section we describe an approximation algorithm for MTO on stars that will also serve as a building block in our approximation algorithms for general trees.

Lemma 2. *If T is a star then at least $1/4$ of all pairs can be satisfied.*

Proof. Choose a random orientation. Each pair is then satisfied with a probability of at least $1/4$. ■

It is easy to use the method of conditional expectations to obtain a deterministic linear time algorithm that produces an orientation of a star that satisfies at least $1/4$ of the pairs. This immediately gives us a $1/4$-approximation algorithm for the problem. As the MTO problem on stars is equivalent to the Max-DiCut problem, an 0.874-approximation for the problem can be obtained using the semidefinite programming based approximation algorithms [2, 9].

Approximating MTO on Caterpillars

Recall that a *caterpillar* is a graph in which all vertices are on a central path or at most one edge away from it. MTO is NP-complete even for caterpillars with maximum degree 3 (the proof is similar to the proof for binary trees, and is omitted for lack of space). We show the following:

Lemma 3. *Let T be a caterpillar. At least $1/8$ of all pairs can be satisfied.*

Proof. Partition edges into 'path' edges which lie on the caterpillar path, and 'bush' edges which "stick" from it. Direct the path edges in a single direction by choosing one of the two at random. Also, randomly direct each of the bush edges. Note that each pair of vertices (u, v) is connected by at most two bush edges and a sub-path. Therefore, the probability that (u, v) is satisfied by the random assignment is at least $1/8$. The claim follows. ■

3.4 Approximating MTO on bounded-depth trees

In this section we present approximation algorithms for rooted, bounded-depth trees that make use of the approximation algorithm for stars. All the results can be extended to unrooted, bounded-diameter trees by rooting them at a "central" vertex so that their depth is bounded by roughly half the diameter.

Consider a tree T with d levels (and depth $d-1$). We denote the vertices at level i by L_i, starting from the root at level 1. For a node v, denote by T_v the subtree rooted at v. Two notions of separation will be useful to us in the approximation algorithms that we design in the sequel.

Definition 2. *A node w in a tree* separates *a pair (u, v), if w is on the path between u and v. w is called the* lowest common ancestor (LCA) *of u and v if in addition it lies on the lowest possible level in the tree.*

Lemma 4. *For any vertex v in T, at least $1/4$ of the pairs separated by v can be satisfied.*

Proof. Re-root the tree at v, and denote its subtrees by $T_1, \ldots, T_l$. We will direct all edges in T_i either toward v or away from v, consistent with the edge between v and T_i. To direct the edges between v and T_i, construct an instance of MTO on a star T' as follows: the root is v, and there are l leaves $v_1, \ldots, v_l$. For each (u, w) separated by v, where $u \in T_x, w \in T_y$, add (v_x, v_y) to P'. By Lemma 2, $1/4$ of the pairs in P' can be satisfied. The edge directions in T' are used to direct the edges of the tree. It is easy to see that any pair (u, w) separated by v, where $u \in T_x, w \in T_y$, is satisfied iff (v_x, v_y) is satisfied in T'. The claim follows. ■

Corollary 3. *For any vertex v in T, at least $1/4$ of the pairs whose LCA is v can be satisfied.*

Definition 3. *For a tree T rooted at a vertex v, we denote by StarMTO(T, P, v) the star-based solution of the MTO problem on T, as described in the proof of Lemma 4.*

Lemma 5. *Let T be a rooted tree with d levels. At least $1/(4d)$ of the pairs in P can be satisfied.*

Proof. As there are d levels, there must be a level j that contains at least $|P|/d$ LCAs of the pairs in P. Compute StarMTO(T_v, P, v) for each node $v \in L_j$. By Corollary 3, at least $|P|/(4d)$ of the pairs are satisfied. ■

The above lemma provides us with a lower bound on the number of pairs that can be satisfied. It implies an approximation algorithm to MTO with a ratio of $1/(4d)$, but the latter ratio can be improved as we show next:

Lemma 6. *Let T be a tree with d levels. MTO can be approximated to within a factor of $1/(2d)$ on T.*

Proof. We form a d-partite graph G_d, in which each node corresponds to a pair in P. The i-th layer is the set of pairs whose LCAs lie on L_i. We connect two vertices (in two layers) by an edge if the two pairs cannot be simultaneously satisfied. Clearly, the maximum number of pairs that can be satisfied is no more than a maximum sized independent set I in G_d.

For the algorithm, find an independent set I' in G_d. Next, solve StarMTO starting with the root level of the tree, and going down the tree. Specifically, for each vertex $v \in L_i$, solve StarMTO(T_v, I', v) on the pairs in the independent set I'. Note that some of the edge directions have been pre-set by previous levels. However, as I' is an independent set, the edge directions set by previous levels do not contradict any of the pairs in the current level. By Lemma 2, at least $|I'|/4$ pairs are satisfied. The approximation ratio is therefore

$$\frac{|I'|}{4|I|} = \frac{\alpha_d}{4}$$

where $\alpha_d = 2/d$ is the approximation ratio achievable for independent sets on a d-partite graph [7]. Overall we get an approximation ratio of $1/(2d)$. ∎

Lemma 6 implies a $1/(2 \lg n)$ approximation algorithm for a complete binary tree, as it contains $\lg n$ levels.

Approximating MTO on General Trees

Lemma 7. *In every tree of size n there is a* centroid *node whose removal breaks the tree into components of size at most $n/2$.*

Our approximation algorithm for general trees is as follows:

MTO(T, P)

1. Find a centroid v, with resulting subtrees $T_1, \ldots, T_l$.
2. Let $A_s = \text{StarMTO}(T, P, v)$.
3. For all $1 \leq j \leq l$, let $P_j = \text{MTO}(T_j, P)$.
4. return $\max\{A_s, \sum_j P_j\}$.

Theorem 4. *Let T be a tree with n nodes. For any set P, MTO(T, P) finds an orientation that satisfies at least $1/(4 \lg n)$ of the pairs.*

Proof. Let $R(n)$ be a lower bound on the fraction of the pairs satisfied by the orientation produced by the algorithm when run on a tree with n vertices. We show by induction that $R(n) \geq \frac{1}{4 \lg n}$.

At the base of the induction $n = 2$. In this case, at least $1/2$ of the pairs are satisfied, and $R(n) \geq 1/2$. Suppose now that $R(k) \geq 1/(4 \lg k)$ for any $k < n$. Let $P(n) = |P|R(n)$ denote the minimum number of pairs satisfied by running MTO on an input of size n. Also, let A be the subset of pairs separated by

the centroid v. By the induction assumption, $\sum_j P(n_j) \geq \frac{|P|-|A|}{4\lg(n/2)}$. Therefore, applying to the recursion

$$P(n) = \max\left\{\frac{|A|}{4}, \sum_j P(n_j)\right\} \geq \max\left\{\frac{|A|}{4}, \frac{|P|-|A|}{4\lg(n/2)}\right\}$$

The two sub-expressions are balanced for $|A| = |P|/\lg n$, implying that

$$R(n) = \frac{P(n)}{|P|} \geq \frac{1}{4\lg n}$$

■

4 Applications to Simulated and Real Network Data

We implemented the general approximation algorithm described above and tested it on simulated and real network data. To evaluate its performance we also implemented the integer-program algorithm which provides an optimal solution to MTO.

4.1 Performance on Simulated Data

As a first test of the algorithm, we measured its approximation ratio performance on random trees, by comparing the solutions it obtained to those obtained by the IP algorithm. The random trees contained 1,000 vertices and were generated by iteratively adding a vertex, and connecting it to one of the already existing vertices uniformly at random. The cause-effect pairs were generated by drawing vertices from the tree uniformly at random.

We tested the algorithm's performance when varying the number of cause-effect pairs from 20 to 1000. The percentage of satisfied pairs along with the implied approximation ratio are displayed in Figure 2. Evidently, the algorithm attains very high approximation ratios in practice reaching up to 0.9 and higher ratios on instances with 500 or more pairs. Notably, even on random instances for which at least 80% of the pairs could be satisfied (see a detailed description in the next paragraph) the average approximation ratio was 0.66 – much higher than the theoretical guarantee.

To test the utility of the algorithm in predicting edge directions, we simulated input with known edge directions as follows: we generated 20-1,000 pairs of vertices. 80% of the pairs were generated in a way that all could be satisfied simultaneously. The other 20% of the pairs were generated randomly, to simulate noise. We randomly chose 50 edges on the paths of the "correct" pairs, and tested the algorithm's accuracy in predicting their direction. The accuracy did not seem to depend on the number of pairs, and was 0.76 on average.

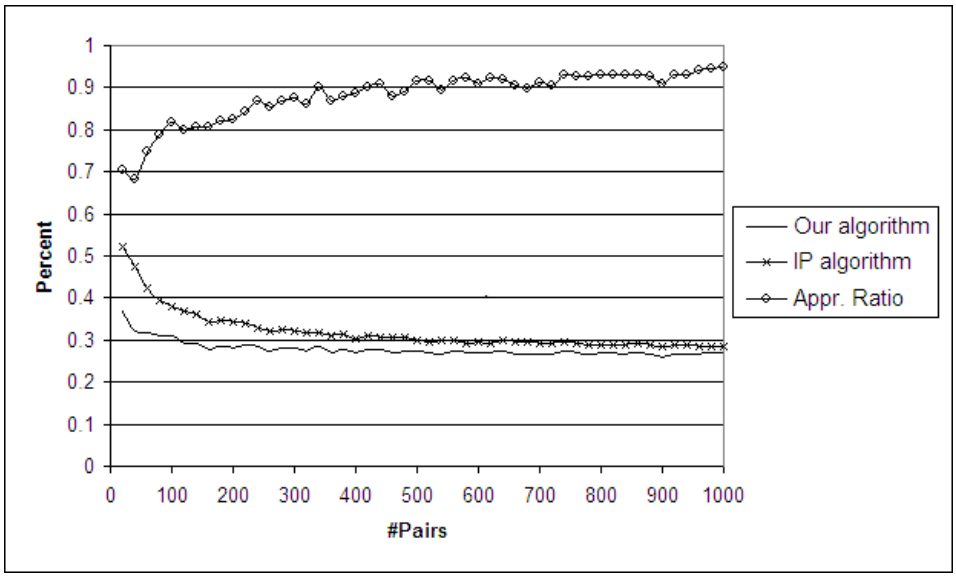

Fig. 2. Performance on simulated data. Percents of satisfied pairs are displayed for both an optimal solution (based on IP) and the approximation algorithm's solution. A third plot depicts the implied approximation ratio.

4.2 Biological Data

Next, we tested the performance of our algorithm on real data. To this end, we used a yeast protein-protein interaction (PPI) network consisting of 15,147 protein-protein interactions obtained from the Database of Interacting Proteins [11]. We complemented this network by additional 596 (kinase-substrate) PPIs from [10] for which the direction of signal flow is known (from the kinase to the substrate), represented as undirected edges in the constructed network. For cause-effect pairs, we used knockout data obtained from [12]. The data set contained 24,457 pairs of a knocked out gene (cause) and an affected gene (effect), out of which 14,295 are pairs of proteins from the network.

After removal of small disconnected components (of size ≤ 3) and cycle-contraction, we obtained a tree with 2,027 vertices and 3,370 cause-effect pairs. Interestingly, about 90% of the vertices in the contracted tree were aligned in a star form. Applying our approximation algorithm to the tree yielded an orientation that satisfied 3,262 of the 3,370 pairs. The optimal solution, obtained using integer programming, satisfied 3,295 pairs, implying a practical approximation ratio of 0.99. This tight ratio matches the ratios observed in the simulations (Figure 2).

The orientation produced by the algorithm provided predictions for 3,880 interaction directions. 148 of these interactions were from the kinase-substrate data set and, hence, their true directions were known. Remarkably, 147 of these

148 directions were predicted correctly. Notably, none of the kinase-substrate interactions were also cause-effect pairs, but rather lied on paths connecting such pairs.

5 Conclusions

In this paper we have studied the problem of orienting a graph so as to satisfy a maximum number of ordered pairs. We have given exact and approximate algorithm to certain restrictions of the problem, and an $O(\log n)$ approximation algorithm for the general case. The algorithm was shown to yield very tight approximation ratios in practice, and attained remarkable accuracy in predicting edge directions on a real protein network.

Several open problems that require further investigation include: (i) closing the gap between the guaranteed approximation ratio in the general case and the approximation hardness result; (ii) tackling the graph orientation problem when some of the edge directions are pre-set (in the biological context this happens when there is prior biological knowledge on directionality or when considering other types of interactions such as transcriptional regulatory ones); and (iii) improving the lower bound on the optimum number of pairs that can be satisfied.

6 Acknowledgments

We thank Richard Karp, Eran Halperin, Tomer Shlomi and Eytan Ruppin for stimulating discussions about the orientation problem. We thank Oved Ourfali for his help with the implementation. We thank Andreas Beyer and Silpa Suthram for providing us with the kinase-substrate data. This work was supported by a grant from the Israel Science Foundation (grant no. 385/06).

References

1. E. M. Arkin and R. Hassin. A note on orientations of mixed graphs. *Discrete Applied Mathematics*, 116(3):271–278, 2002.
2. U. Feige and M. X. Goemans. Aproximating the value of two prover proof systems, with applications to MAX 2-SAT and MAX DI-CUT. In *ISTCS*, pages 182–189, 1995.
3. S. Fields. High-throughput two-hybrid analysis. the promise and the peril. *Febs J*, 272(21):5391–5399, 2005.
4. S. L. Hakimi, E. Schmeichel, and N. E. Young. Orienting graphs to optimize reachability. *Information Processing Letters*, 63(5):229–235, 1997.
5. R. Hassin and N. Megiddo. On orientations and shortest paths. *Linear Algebra and its applications*, 114/115:589–602, 1989.
6. J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.
7. D. S. Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Applied Mathematics*, 6:243–254, 1983.

8. V. Kann, J. Lagergren, and A. Panconesi. Approximability of maximum splitting of k-sets and some other apx-complete problems. *Information Processing Letters*, 58(3):105–110, 1996.
9. M. Lewin, D. Livnat, and U. Zwick. Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems. In *Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 67–82, London, UK, 2002. Springer-Verlag.
10. J. Ptacek, G. Devgan, G. Michaud, H. Zhu, X. Zhu, J. Fasolo, H. Guo, G. Jona, A. Breitkreutz, R. Sopko, R. R. McCartney, M. C. Schmidt, N. Rachidi, S. J. Lee, A. S. Mah, L. Meng, M. J. Stark, D. F. Stern, C. De Virgilio, M. Tyers, B. Andrews, M. Gerstein, B. Schweitzer, P. F. Predki, and M. Snyder. Global analysis of protein phosphorylation in yeast. *Nature 438*, pages 679–84, 2005.
11. L. Salwinski, C. S. Miller, A. J. Smith, F. K. Pettit, J. U. Bowie, , and D. Eisenberg. The database of interacting proteins: 2004 update. *Nucleic Acids Research, 32*, page D449, 2004.
12. C.-H. Yeang, T. Ideker, and T. Jaakkola. Physical network models. *Journal of Computational Biology*, 11(2/3):243–262, 2004.